Contents

Part I

Simulation Concepts

Chapter 1

Introduction to Simulation

Simulation has been in use for over 40 years, but rather than being "over the hill," it's just moving into its prime. Gartner (`www.gartner.com`) is a leading provider of technical research and advice for business. In a recent analysis, Gartner [11] identified the top ten strategic technologies for 2010 and ranked *Advanced Analytics*, including simulation, as number two:

> "Optimization and simulation is using analytical tools and models to maximize business process and decision effectiveness by examining alternative outcomes and scenarios, before, during and after process implementation and execution. This can be viewed as a third step in supporting operational business decisions. Fixed rules and prepared policies gave way to more informed decisions powered by the right information delivered at the right time, whether through customer relationship management (CRM) or enterprise resource planning (ERP) or other applications. The new step is to provide simulation, prediction, optimization and other analytics, not simply information, to empower even more decision flexibility at the time and place of every business process action. The new step looks into the future, predicting what can or will happen."

Simulation-related advancements in hardware and software over the last decade have been dramatic. Powerful personal computers now provide processing power unheard of even a few years ago. Advances in user interfaces and product design have resulted in software that's significantly easier to use, lowering the expertise required to use simulation effectively. Breakthroughs in object-oriented technology provide significantly improved modeling flexibility and allow accurate modeling of highly complex systems. Hardware, software, and publicly available symbols allow even novices to produce simulations with compelling 3D animation to support communication between people of all backgrounds. These innovations and other developments are working together to propel simulation into a new position as a critical technology.

In this book we hope to open up the world of simulation to you, providing exposure to general simulation technology and success skills, as well as a practical introduction to a state-of-the-art simulation package.

1.1 About the Book

This book is divided into three parts. The first part, **Simulation Concepts**, encompasses chapters 1–4. This part is intended to provide a sound basis in the underlying concepts before introducing any product-specific concepts. Chapter 1, *Introduction to Simulation*, covers typical simulation applications, how to identify an appropriate simulation application, and how to carry out a simulation project. Chapter 2, *Basics of Queueing Theory*, introduces the concepts of queueing theory, its strengths and limitations, and in particular how it can be used to help validate components of later simulation modeling. Chapter 3, *Approaches to Simulation*, introduces some of the technical aspects and terminology of simulation, delineates the different kinds of simulation, then illustrates this by working through a manual simulation and two spreadsheet simulations. Chapter 4, *Input Analysis*, discusses different types of inputs to simulations, methods for converting observed real-world data into something useful to a simulation project, and generating the appropriate random quantities needed in most simulations.

The second part of the book, **Simulation Modeling with Simio**, is composed of six chapters. This part introduces more detailed simulation concepts illustrated with numerous examples implemented in Simio. Rather than breaking up the technical components (like validation, and output analysis) into separate chapters, we look at each example as a mini project and introduce successively more concepts with each project. This approach provides the opportunity to learn the best overall practices and skills at an early stage, and then reinforce those skills with each successive project.

Chapter 5, *First Simio Models*, starts with a brief overview of Simio itself, and then directly launches into building a single-server queueing model in Simio. The primary goal of this chapter is to introduce the simulation model-building process using Simio. While the basic model-building and analysis processes themselves aren't specific to Simio, we'll focus on Simio as an implementation vehicle. This process not only introduces modeling skills, but also covers the statistical analysis of simulation output results, experimentation, and model verification. That same model is then reproduced using lower-level tools to illustrate another possible modeling approach, as well as to provide greater insight into what's happening "behind the curtain." The chapter continues with a third, more interesting model of an ATM machine and introduces additional output analysis using Simio's innovative SMORE plots. The chapter closes with some additional discussion of output analysis outside of Simio, as well as adding basic 3D animation to the model.

The goal of Chapter 6, *Intermediate Modeling with Simio*, is to build on the basic Simio modeling-and-analysis concepts presented earlier so that we

can start developing and experimenting with models of more realistic systems. We'll start by discussing a bit more about how Simio works and its general framework. Then we'll build an electronics-assembly model and successively add additional features, including modeling multiple processes, conditional branching and merging, etc. As we develop these models, we'll continue to introduce and use new Simio features. We'll also resume our investigation of how to set up and analyze sound statistical simulation experiments, this time by considering the common goal of comparing multiple alternative scenarios. By the end of this chapter, you should have a good understanding of how to model and analyze systems of intermediate complexity with Simio.

Chapter 7, *Working with Model Data*, takes a wider view and examines the many types of data that are often required to represent a real system. We'll start by building a simple emergency-department (ED) model, and will show how to meet its input-data requirements using Simio's data-table construct. We'll successively add more detail to the model to illustrate the concepts of sequence tables, relational data tables, arrival tables, and importing and exporting data tables. We'll continue enhancing the ED model to illustrate work schedules, rate tables, and function tables. The chapter ends with a brief introduction to lists, arrays, and changeover matrices. After completing this chapter you should have a good command of the types of data frequently encountered in models, and the Simio choices for representing those data.

Animation and Entity Movement, Chapter 8, discusses the enhanced validation, communication, and credibility that 2D and 3D animation can bring to a simulation project. Then we explore the various animation tools available, including background animation, custom symbols, and status objects. We'll revisit our previous electronics-assembly model to practice some new animation skills, as well as to explore the different types of links available, and add conveyors to handle the work flow. Finally, we'll introduce the Simio Vehicle and Worker objects for assisted entity movement, and revisit our earlier ED model to consider staffing and improve the animation.

Chapter 9 is *Advanced Modeling with Simio*. We start with a simpler version of our ED model, with the goal of demonstrating the use of models for decision-making, and in particular simulation-based optimization. Then we'll introduce a new pizza-shop example to illustrate a few new modeling constructs, as well as bring together concepts that were previously introduced. A third and final model, an assembly line, allows study of buffer-space allocation to maximize throughput.

Chapter 10, *Customizing and Extending Simio* starts with some slightly more advanced material — it builds on the prior experience using add-on processes to provide guidance in building your own custom objects and libraries. It includes examples of building objects hierarchically from base objects, and sub-classing standard library objects. This chapter ends with an introduction to Simio's extendability through programming your own rules, components, and add-ons to Simio.

The third and final part of the book is **Case Studies Using Simio**. Chapter 11, *Introductory Cases*, provides four somewhat small but realistic case

studies to allow you to practice your modeling skills. Chapter 11, *Advanced Cases*, provides four larger and more challenging case studies for more advanced study.

1.2 Systems and Models

A *System* is a very broad term used to describe a set of related components that together work toward some purpose. A system might be something as simple as a waiting line at an automated teller machine (ATM), or as complex as a complete airport or a worldwide distribution network. In any such system, be it existing or merely contemplated, it's natural and sometimes even essential to understand how it will behave and perform under various configurations and circumstances.

If the system already exists, sometimes you can gain the necessary understanding by careful observation. One drawback of this approach is that you may need to watch the real system a long time in order to observe the particular conditions of interest even once, let alone making enough observations to reach reliable conclusions. And of course, for some systems (say that worldwide distribution network), it's hard to find the vantage point from which you can observe the entire system at an adequate level of detail. Additional problems arise when you want to study changes to the system. In some cases it may be easy simply to make the change in the real system — for example, add a temporary second person to a shift to observe the impact. But in many cases this is simply not practical — consider the investment required to evaluate whether you should use a standard machine that costs $300,000 or a high-performance machine that costs $400,000. And finally, if the real system doesn't yet exist, no observation at all is possible.

For all the reasons above, we often choose to use some sort of model to gain understanding. There are many types of models, each with their own advantages and limitations. *Physical models*, such as a model of a car or airplane, can provide both a sense of reality as well as interaction with the physical environment, as in wind-tunnel testing. There are many different types of *analytical models* that use mathematical representations to facilitate understanding — these can be quite good in specific problem domains, but available domains are often limited. Simulation is yet another modeling approach that has much broader applicability.

Computer Simulation is the imitation of the operation of a system and its internal processes, over time, and in appropriate detail to draw conclusions about the system's behavior. Simulation models are usually created using software designed to represent common system components, their behavior, and their time-based interactions, and to record an artificial "history" of a model run as well as summaries and inferences about system characteristics. Simulation is often used for both predicting the effect of changes to existing systems, as well as predicting the performance of new systems. Simulations are frequently used in the design, emulation, and operation of systems.

Simulations may be Stochastic or Deterministic. In a *Stochastic* simulation (the most common), randomness is introduced to represent the variation found in most systems. For example, the results of activities involving people (time to complete a task, quality level) always vary, external inputs (customers, materials) vary, and exceptions (failures) occur. *Deterministic* models have no variation. These are rare in design applications, but more common in model-based decision support such as scheduling and emulation applications. Section 3.1.3 discusses this further.

There are two main types of simulation, *Discrete* and *Continuous*. The terms discrete and continuous refer to the changing nature of the states that describe the system. Some states (e.g., the length of a queue, status of a worker) change at discrete points in time (called *event times*). Other states (e.g., pressure in a tank, temperature in an oven) change continuously over time. Some systems are pure discrete or continuous, while others have both types of states present. In Section 3.1.2 we discuss this further, and give an example of a continuous simulation.

Continuous systems are defined by *differential equations* that specify the rate of change — simulation software uses numerical integration to generate a solution for the differential equations over time. *System Dynamics* is a graphical approach for creating simple models using the same underlying concept, and is often used to model population dynamics, market growth/decay, etc.

There are four discrete modeling paradigms that have evolved over time. *Events* model the points in time when the system state can change (e.g., a customer arrival, or departure). *Processes* model a sequence of actions that take place over time (a part in a manufacturing system seizes a worker, delays by a service time, then releases the worker). *Objects* describe the model from the point of view of the facility. *Agent Based Modeling* (ABM) is a special case of objects — the system behavior emerges from the interaction of a large number of autonomous intelligent objects (populations, soldiers, markets, etc.). The distinction between these paradigms is somewhat blurred because some modern packages incorporate multiple paradigms. Simio is a multi-paradigm modeling tool that combines all these paradigms into a single framework. You can use a single paradigm, or combine multiple paradigms in the same model. Simio combines the ease of objects with the flexibility of processes.

Simulation has been applied to a huge variety of settings. Here are just a few samples of areas where simulation has been used to understand and improve the system effectiveness:

Airports: Parking-lot shuttles, ticketing, security, terminal transportation, food court, baggage handling, gate assignment, plane deicing.

Hospitals: Emergency department, disaster planning, ambulance dispatching, regional service strategies, resource allocation.

Ports: Inbound traffic, outbound traffic, port management, container storage, capital investments, crane operation.

Mining: Material transfer, labor transportation, equipment allocation, bulk material mixing.

Amusement Parks: Guest transportation, ride design/startup, waiting lines, ride staffing, crowd management.

Call Centers: Staffing, skill-level assessment, service improvement, training plans, scheduling algorithms.

Supply Chains: Risk reduction, reorder points, production allocation, inventory positioning, transportation, growth management, contingency planning.

Manufacturing: Capital-investment analysis, line optimization, product-mix changes, productivity improvement, transportation, labor reduction.

Military: Logistics, maintenance, combat, counterinsurgency, search and detection, humanitarian relief.

Telecommunications: Message transfer, routing, reliability, network robustness to outages or attacks.

Criminal Justice System: Probation/parole operations, prison utilization and capacity.

Emergency-Response System: Response time, station location, equipment levels, staffing.

Public-Sector: Allocation of voting machines to precincts.

Customer Service: Direct-service improvement, back-office operations, resource allocation, capacity planning.

Some people still think of simulation as a tool only for manufacturing, but that's obviously not the case. The domains and applications of simulation are wide-ranging and virtually limitless.

1.3 When to Simulate (and When Not To)

Simulation of complicated systems has become quite popular. One of the main reasons for this is embodied in that word "complicated." If the system of interest were actually simple enough to be validly represented by an exact analytical model, simulation wouldn't be needed, and indeed shouldn't be used: we should instead use such exact analytical methods like queueing theory, probability, or maybe even just simple algebra or calculus. Simulating a simple system for which we can find an exact analytical solution can only add noise, i.e. uncertainty, to the results, making them less precise.

But in reality we quickly get out of the realm of such very "simple" models since, well, the world tends to be a complicated place. And if we're serious

about building a *valid* model of a complicated system, that model will itself likely be fairly complicated and not amenable to a simple analytical analysis. We could go ahead and build a simple model of a complicated system with the goal of preserving our ability to get an exact analytical solution, but the resulting model would probably be *overly* simple (simplistic, even), and we'd be left wondering if such a model validly represents the system. We may be able to get a nice, clean, exact, closed-form analytical solution to our simple model, but because we probably made a lot of simplifying assumptions (some of which might be quite questionable in reality) to get to our analytical model, it's hard to say of what, exactly, in reality do we have a solution — sure, it's a solution to the *model*, but that model might not bear much resemblance to *reality*.

And it's hard, maybe impossible, to quantify or measure just *how* unrealistic a model is; it's not even clear that asking such a question means much. On the other hand, if we don't concern ourselves with building a model that will have an analytical solution in the end, we're freed up to allow things in the model to become as complicated and messy as they need to be in order to mimic the system in a valid way. But, a nice simple analytical model is no longer available, so we need to turn to simulation, where we simply mimic the complicated system, via its complicated (but realistic) model, numerically on a computer, and watch what happens to the results. Part of this is usually to allow some model inputs to be *stochastic*, i.e., random and represented by "draws" from probability distributions rather than by fixed constant input values, if such is the way things are in reality, and this causes the results from our simulation model to be likewise stochastic, and thus uncertain.

Clearly, this uncertainty or imprecision in simulation output is a downside. But, as we'll, see, it's *not* hard to measure the degree of this imprecision, and if we don't like the answer (i.e., the results are too imprecise), we know how to deal with it, and can indeed do so, often by just simulating some more and consuming more computer time. Unlike most statistical sampling experiments, we're in complete control of of the "randomness" and numbers of replications, and can use this control to gain any level of precision that we like. It used to be that computer time was a real barrier to simulation's utility, and in some cases it still can be for tremendously complex models that require hours or maybe days to run once. But for many simulation models, it's now possible with readily available (and relatively cheap) computing power to do enough simulating to get results with imprecision that's both measurable and acceptably small — and with no gnawing doubt about whether our model is so simple as to be unrealistic and not valid.

In years gone by, simulation has sometimes been regarded as "the method of last resort," or an approach to be taken only "when all else fails" ([48], pp. 887, 890), and as we noted above, we agree that simulation should not be used if a valid analytical model is available. But in many (perhaps most) cases, the actual system is just too complicated or does not obey the rules to allow for an analytical model of any credible validity to be built and analyzed. In our opinion, it's better to simulate the right model and get an approximate answer

whose imprecision can be dealt with, than to do an exact analytical analysis of the wrong model and get an answer whose error cannot be even be quantified, a situation that's worse than imprecision.

While we are talking about precise answers, the examples and figures in this text edition were created with Simio Version 3.44. Because each version of the software may contain changes that could effect low-level behavior (like the processing order of simultaneous events) different versions could produce different output results for an interactive run. You may wonder "Which results are correct?" Each one is as correct (or as incorrect) as the others! In this book you'll learn how to create statistically valid results, and how to recognize when you have (or don't have) them. With the possible exception of a rare bug fix between versions, every version should generate the same statistically valid results for the same model.

1.4 Simulation Success Skills

Learning to use a simulation tool and understand the underlying technology will not guarantee your success. Conducting successful simulation projects requires much more than that. Newcomers to simulation often inquire how they can be successful in simulation. The answer is easy: "Work hard and do everything right." But perhaps you want a bit more detail. Let's identify some of the more important issues that should be considered.

1.4.1 Project Objectives

The first question to ask when presented with a simulation is "What's your objective?" Although it may seem like an obvious question with a simple answer, it often happens that stakeholders don't know the answer. Many projects start with a fixed deliverable date, but often only a rough idea of what will be delivered and a vague idea of how it will be done.

Your first role may be to help clarify the objectives. But before you can help with objectives, you need to get to know the stakeholders. A *stakeholder* is someone who commissions, funds, uses, or is affected by the project. Some stakeholders are obvious — your boss is likely to be stakeholder (if you're a student, your instructor is most certainly a stakeholder). But sometimes you have to work a bit to identify all the key stakeholders. Why should you care? In part because stakeholders usually have differing (and conflicting) objectives.

Let's say that you're asked to model a specific manufacturing facility at a large corporation, and evaluate whether a new 4 million dollar crane will provide the desired results (increases in product throughput, decreases in waiting time, reductions in maintenance, etc.). Here are some possible stakeholders and what their objectives might be in a typical situation:

- Manager of industrial engineering (IE) (your boss): She wants to prove that IE adds value to the corporation, so she wants you to demonstrate dramatic cost savings or productivity improvement. She also wants a

nice 3D animation she can use to market your services elsewhere in the corporation.

- Production Manager: He's convinced that buying a new crane is the only way he can meet his production targets, and has instructed his key people to provide you the information to help you prove that.

- VP-Production: He's been around a long time and is not convinced that this "simulation" thing offers any real benefit. He's marginally supporting this effort due to political pressure, but fully expects (and secretly hopes) the project will fail.

- VP-Finance: She's very concerned about spending the money for the crane, but is also concerned about inadequate productivity. She's actually the one who, in the last executive meeting, insisted on commissioning a simulation study to get an objective analysis.

- Line Supervisor: She's worked there 15 years and is responsible for material movement. She knows that there are less-expensive and equally effective ways to increase productivity, and would be happy to share that information if anyone bothered to ask her.

- Materials Laborer: Much of his time is currently spent moving materials, and he's afraid of getting laid off if a new crane is purchased. So he'll do his best to convince you that a new crane is a bad idea.

- Engineering Manager: His staff is already overwhelmed, so he doesn't want to be involved unless absolutely necessary. But if a new crane is going to be purchased, he has some very specific ideas of how it should be configured and used.

This scenario is actually a composite of some real cases. Smaller projects and smaller companies might have fewer stakeholders, but the underlying principles remain the same. Conflicting objectives and motivations are not at all unusual. Each of the stakeholders has valuable project input, but it's important to take their biases and motivations into account when evaluating their input.

So now that we've gotten to know the stakeholders a bit, we need to determine how each one views or contributes to the project objectives and attempt to prioritize them appropriately. In order to identify key objectives, you must ask questions like these:

- What do you want to evaluate, or hope to prove?

- What's the model scope? How much detail is anticipated for each component of the system?

- What components are critical? Which less-important components might be approximated?

- What input data can be made available, how good are they, who will provide them, and when?
- How much experimentation will be required? Will optimum-seeking be required?
- How will any animation be used (animation for validation is quite different than animation presented to a board of directors)?
- In what form do you want results (verbal presentation, detailed numbers, summaries, graphs, text reports)?

One very good way to help identify clear objectives is to design a mock-up of the final report. You can say, *"If I generate a report with the following information in a format like this, will that address your needs?"* Once you can get general agreement on the form and content of the final report, you can often work backwards to determine the appropriate level of detail and address other modeling concerns. This process can also help bring out unrecognized modeling objectives.

Sometimes the necessary project clarity is not there. If so, and you go ahead anyway to plan the entire project including deliverables, resources, and date, you're setting yourself up for failure. Lack of project clarity is a clear call to do the project in phases. Starting with a small prototype will often help clarify the big issues. Based on those prototype experiences, you might find that you can do a detailed plan for subsequent phases. We'll talk more about that in the next section.

1.4.2 Functional Specification

> *"If you don't know where you're going,*
> *how will you know when you get there?"*
>
> *Carpenter's advice: "Measure twice. Cut once."*

If you've followed the advice from the Section 1.4.1, you now have at least some basic project objectives. You're ready to start building the model, right? Wrong! In most cases your stakeholders will be looking for some commitments.

- When will you get it done (is yesterday too soon)?
- How much will it cost (or how many resources will it require)?
- How comprehensive will the model be (or what specific system aspects will be included)?
- What will be the quality (or how will it be verified and validated)?

Are you ready to give reliable answers to those questions? Probably not.

Of course the worst possible, but quite common, situation is that the *stakeholder* will supply answers to all of those questions and leave it to you to deliver.

Picture a statement like "I'll pay you $5000 to provide a thorough, validated analysis of ... to be delivered 5 days from now." If accepted, such a statement often results in a lot of overtime to produce a partially complete, unvalidated model that's a week or two late. And as for the promised money ... well, the customer didn't get what he asked for, now, did he?

It's OK for the customer to specify answers to *two* of those questions, and in rare cases maybe even *three*. But you must reserve the right to adjust at least one or two of those answers. You might cut the scope to meet a deadline. Or you might extend the deadline to achieve the scope. Or, you might double both the resources and the cost to achieve the scope and meet the date (adjusting the quality is seldom a good idea).

If you're fortunate, the stakeholder will allow you to answer all four questions (of course, reserving the right to reject your proposal). But how do you come up with good answers? By creating a *functional specification*, which is a document describing exactly what will be delivered, when, how, and by whom. While the details required in a functional specification vary by application and project size, typical components may include:

1. Introduction

    a) Simulation Objectives: Discussion of high-level objectives. What's the desired outcome of this project?

    b) Identification of Stakeholders: Who are the primary people concerned with the results from this model? Which other people are also concerned? How will the model be used and by whom? How will they learn it?

2. System Description and Modeling Approach: Overview of system components and approaches for modeling them. Including, but not limited to, the following components:

    a) Equipment: Each piece of equipment should be described in detail, including its behavior, setups, schedules, reliability, and other aspects that might affect the model. Include data tables and diagrams as needed. Where data do not yet exist, they should be identified as such.

    b) Product Types: What products are involved? How do they differ? How do they relate to each other? What level of detail is required for each product or product group?

    c) Operations: Each operation should be described in detail including its behavior, setups, schedules, reliability, and other aspects that might affect the model. Include data tables and diagrams as needed. Where data do not yet exist, they should be identified as such.

    d) Transportation: Internal and external transportation should be described in adequate detail.

3. Input Data: What data should be considered for model input? Who will provide this information? When? In what format?

4. Output Data: What data should be produced by the model? In this section, a mock-up of the final report will help clarify expectations for all parties.

5. Project Deliverables: Discuss all agreed-upon project deliverables. When this list is fulfilled, the project is deemed complete.

   a) Documentation: What model documentation, instructions, or user manual will be provided? At what level of detail?

   b) Software and Training: If it's intended that the user will interact directly with the model, discuss the software that's required, what software, if any, will be included in the project price quote, and what, if any, custom interface will be provided. Also discuss what project or product training is recommended or will be supplied.

   c) Animation: What are the animation deliverables and for what purposes will the animations be used (model validation, stakeholder buy-in, marketing)? 2D or 3D? Are existing layouts and symbols available, and in what form? What will be provided, by whom, and when?

6. Project Phases: Describe each project phase (if more than one) and the estimated effort, delivery date, and charge for each phase.

7. Signoffs: Signature section for primary stakeholders.

At the beginning of a project there's a natural inclination just to start modeling. There's time pressure. Ideas are flowing. There's excitement. It's very hard to stop and do a functional specification. But trust us on this — *doing a functional specification is worth the effort.* Look back at those quotations at the beginning of this section. Pausing to determine where you're going and how you're going to get there can save misdirected effort and wasted time. We recommend that approximately the first 10% of the total estimated project time be spent on creating a prototype and a functional specification. Yes, that means if you expect the project may take 20 days, you should spend about two days on this. As a result, you may well find that the project will require 40 days to finish — certainly bad news, but much better to find out up front while you still have time to consider alternatives (reprioritize the objectives, reduce the scope, add resources, etc.).

1.4.3 Project Iterations

Simulation projects are best done as an iterative process. Even from the first steps. You might think you could just define your objectives, create a functional specification, and then create a prototype. But while you're writing

the functional specification, you'll likely discover new objectives. And while you're doing the prototype, you'll discover important new things to add to the functional specification.

As you get further into the project, an iterative approach becomes even more important. A simulation novice will often get an idea and start modeling it, then keep adding to the model until it's complete — and *only then* run the model. But even the best modeler, using the best tools, will make mistakes. But when all you know is that your mistake is "somewhere in the model," it's very hard to find it and fix it. Based on our collective experience in teaching simulation, this is a huge problem for students new to simulation.

More experienced modelers will typically build a small piece of the model, then run it, test it, debug it, and verify that it does what the modeler expected it would do. Then repeat that process with another small piece of the model. As soon as enough of the model exists to compare to the real world, then validate, as much as possible, that that entire section of the model matches the intended system behavior. Keep repeating this iterative process until the model is complete. At each step in the process, finding and fixing problems is much easier because it's very likely a problem in the small piece that was most recently added. And at each step you can save under a different name (like `MyModelV1`, `MyModelV2`, or with full dates and even times appended to the file names), to allow reverting to an earlier version if necessary.

Another benefit of this iterative approach, especially for novices, is that potentially-major problems can be eliminated early. Let's say that you built an entire model based on a faulty assumption of how entity grouping worked. And only at the very end did you discover your misunderstanding. At that point it might require extensive rework to change the basis of your model. However, if you were building your model iteratively, you'd probably have discovered your misunderstanding the very first time you used the grouping construct, at which time it would be relatively easy to take a better strategy.

A final, and extremely important benefit of the iterative approach is the ability to prioritize. *For each iteration, work on the most important small section of the model that's remaining.* The one predictable thing about software development of all types is that it almost always takes much longer than expected. Building simulation models often shares that same problem. If you run out of project time when following a non-iterative approach and your model is not yet even working, let alone verified and validated, you essentially have nothing useful to show for your efforts. But if you run out of time when following an iterative approach, you have a portion of the model that's completed, verified, validated, and ready for use. And if you've been working on the highest-priority task at each iteration, you may find that the portion completed is actually enough to fulfill most of the project goals (look up the 80-20 Rule or the Pareto principle to see why).

Although it may vary some by project and application, the general steps in a simulation study are:

1. Define high-level objectives and identify stakeholders.

2. Define the functional specification, including detailed goals, model boundaries, level of detail, modeling approach, and output measures. Design the final report.
3. Build a prototype. Update steps 1 and 2 as necessary.
4. Model or enhance a high-priority piece of the system. Document and verify it. Iterate.
5. Collect and incorporate model input data.
6. Verify and validate the model. Involve stakeholders. Return to step 4 as necessary.
7. Design experiments. Make production runs. Involve stakeholders. Return to step 4 as necessary.
8. Document the results and the model.
9. Present the results and collect your kudos.

As you're iterating, don't waste the opportunity to *communicate regularly with the stakeholders*. Stakeholders don't like surprises. If the project is producing results that differ from what was expected, learn together why that's happening. If the project is behind schedule, let stakeholders know early so that serious problems can be avoided. Don't think of stakeholders as just a client, and certainly not as an adversary. Think of stakeholders as a partner — you can help each other to obtain the best possible results from this project. And those results often come from the detailed system exploration that's necessary to uncover the actual processes being modeled. In fact, in many projects a large portion of the value occurs before any simulation "results" are even generated — due to the knowledge gained from the early exploration by modelers and frequent collaboration with stakeholders.

1.4.4 Project Management and Agility

There are many aspects to a successful project, but one of the most obvious is making the completion deadline. A project that produces results after the decision is made has little value. Other, often-related, aspects are the cost, resources, and time consumed. A project that runs over budget may be canceled before it gets close to completion. You must pay appropriate attention to completion dates and project costs. But both of those are outcomes of how you manage the day-to-day project details.

A well-managed project starts by having clear goals and a solid functional specification to guide your decisions. Throughout the project, you'll be making large and small decisions like:

- How much detail should be modeled in a particular section?
- How much input data do I need to collect?

- To which output data should I pay most attention?
- When is the model considered to be valid?
- How much time should I spend on animation? Analysis?
- What should I do next?

In almost every case, the functional specification should directly or indirectly provide the answers. You've already captured and prioritized the objectives of your key stakeholders. That information should become the basis of most decisions.

One of the things you'll have to prioritize is "evolving specifications" or new stakeholder requests, sometimes called "scope creep." One extreme is to take a hard line and say "if it's not in the functional specification, it's not in the model." While in some rare cases this response may be appropriate and necessary, in most cases it's not. Simulation is an exploratory and learning process. As you explore new areas and learn more about the target system, it's only natural that new issues, approaches, and areas of study will evolve. Refusing to deal with these severely limits the potential value of the simulation (and your value as a solution provider).

Another extreme is to take the approach that the stakeholder is always right, and if she asked you to work on something new, it *must* be the right thing to do. While this response makes the stakeholder happy in the short-term, the most likely longer-term outcome is a late or even unfinished project — and a *very* unhappy stakeholder! If you're always chasing the latest idea, you may never have the time to finish the high-priority work necessary to produce any value at all.

The key is to manage these opportunities — that management starts with open communication with the stakeholders and revisiting the items in the functional specification and their relative priorities. When something is added to the project, something else needs to change. Perhaps addressing the new item is important enough to postpone the project deadline a bit. If not, perhaps this new item is more important than some other task that can be dropped (or moved to the "wish list" that's developed when things go better than expected). Or perhaps this new item itself should be moved to the "wish list."

Our definition of *agility* is the ability to react quickly and appropriately to change. Your ability to be agile will be a significant contributor to your success in simulation.

1.4.5 Stakeholder and Simulationist Bills of Rights

We'll end this chapter with an acknowledgement that stakeholders have reasonable expectations of what you will do for them (Figure 1.1). Give these expectations careful consideration to improve the effectiveness and success of your next project. But along with those expectations stakeholders have some responsibilities to you as well (Figure 1.2). Discussing both sets of these ex-

Simulation Stakeholder Bill of Rights

The people who request, pay for, consume, or are affected by a simulation project and its results are often referred to as its stakeholders. For any simulation project the stakeholders should have reasonable expectations from the people actually doing the work. Here are some basic stakeholder rights that should be assured.

1 **Partnership** – The modeler will do more than provide information on request. The modeler will assume some ownership of helping stakeholders determine the right problems and identify and evaluate proposed solutions.

2 **Functional Specification** – A specification will be created at the beginning of the project to help define clear project objectives, deadlines, data, responsibilities, reporting needs, and other project aspects. This specification will be used as a guide throughout the project, especially when tradeoffs must be considered.

3 **Prototype** – All but the simplest projects will have a prototype to help stakeholders and the modeler communicate and visualize the project scope, approach, and outcomes. The prototype is often done as part of the functional specification.

4 **Level of Detail** – The model will be created at an appropriate level of detail to address the stated objectives. Too much or too little detail could lead to an incomplete, misunderstood, or even useless model.

5 **Phased Approach** – The project will be divided into phases and the interim results should be shared with stakeholders. This allows problems in approach, detail, data, timeliness, or other areas to be discovered and addressed early and reduces the chance of an unfortunate surprise at the end of a project.

6 **Timeliness** – If a decision-making date has been clearly identified, usable results will be provided by that date. If project completion has been delayed, regardless of reason or fault, the model will be re-scoped so that the existing work can provide value and contribute to effective decision-making.

7 **Agility** – Modeling is a discovery process and often new directions will evolve over the course of the project. While observing the limitations of level of detail, timeliness, and other aspects of the functional specification, a modeler will attempt to adjust project direction appropriately to meet evolving needs.

8 **Validated and Verified** – The modeler will certify that the model conforms to the design in the functional specification and that the model appropriately represents the actual operation. If there is inadequate time for accuracy, there is inadequate time for the modeling effort.

9 **Animation** – Every model deserves at least simple animation to aid in verification and communication with stakeholders.

10 **Clear Accurate Results** – The project results will be summarized and expressed in a form and terminology useful to stakeholders. Since simulation results are an estimate, proper analysis will be done so that the stakeholders are informed of the accuracy of the results.

11 **Documentation** – The model will be adequately documented both internally and externally to support both immediate objectives and long term model viability.

12 **Integrity** – The results and recommendations are based only on facts and analysis and are not influenced by politics, effort, or other inappropriate factors.

Note: This is the companion piece to Simulationist Bill of Rights, which outlines reasonable expectations a modeler should have in a simulation project. To read that and more, visit our website.

Simio LLC -- Forward Thinking -- www.simio.com -- © 2010

Figure 1.1: Simulation Stakeholder Bill of Rights.

Simulationist Bill of Rights

The companion ***Simulation Stakeholder Bill of Rights*** proposed some reasonable expectations that a consumer of a simulation project might have. But this is not a one-way street. The modeler or simulationist should have some reasonable expectations as well.

1 **Clear Objectives** – A simulationist can help stakeholders discover and refine their objectives, but clearly the stakeholders must agree on project objectives. The primary objectives must remain solid throughout the project.

2 **Stakeholder Participation** – Adequate access and cooperation must be provided by the people who know the system both in the early phases and throughout the project. Stakeholders will need to be involved periodically to assess progress and resolve outstanding issues.

3 **Timely Data** – The functional specification should describe what data will be required, when it will be delivered and by whom. Late, missing, or poor quality data can have a dramatic impact on a project.

4 **Management Support** – The simulationist's manager should support the project as needed not only in issues like tools and training discussed below, but also in shielding the simulationist from energy sapping politics and bureaucracy.

5 **Cost of Agility** – If stakeholders ask for project changes, they should be flexible in other aspects such as delivery date, level of detail, scope, or project cost.

6 **Timely Review/Feedback** – Interim updates should be reviewed promptly and thoughtfully by the appropriate people so that meaningful feedback can be provided and any necessary course corrections can be immediately made.

7 **Reasonable Expectations** – Stakeholders must recognize the limitations of the technology and project constraints and not have unrealistic expectations. A project based on the assumption of long work hours is a project that has been poorly managed.

8 **"Don't shoot the messenger"** – The modeler should not be criticized if the results promote an unexpected or undesirable conclusion.

9 **Proper Tools** – A simulationist should be provided the right hardware and software appropriate to the project. While "the best and latest" is not always required, a simulationist should not have to waste time on outdated or inappropriate software and inefficient hardware.

10 **Training and Support** – A simulationist should not be expected to "plunge ahead" into unfamiliar software and applications without training. Proper training and support should be provided.

11 **Integrity** – A simulationist should be free from coercion. If a stakeholder "knows" the right answer before the project starts, then there is no point to starting the project. If not, then the objectivity of the analysis should be respected with no coercion to change the model to produce the desired results.

12 **Respect** – A good simulationist may sometimes make the job look easy, but don't take them for granted. A project often "looks" easy only because the simulationist did everything right, a feat that in itself is very difficult. And sometimes a project looks easy only because others have not seen the nights and weekends involved.

Figure 1.2: Simulationist Bill of Rights.

pectations ahead of time can enhance communications and help ensure that your project is successful — a win-win situation that meets everyone's needs. These "rights" were excerpted from the Success in Simulation [46] blog at `www.simio.com/blog` and used with permission. We urge you to peruse the early topics of this non-commercial blog for its many success tips and short interesting topics.